He Who Takes The Credit

Table of Contents

Introduction

In [1]:
```
import numpy as np
import sklearn
from xgboost import XGBClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from time import time
import pandas as pd
from lime.lime_tabular import LimeTabularExplainer
from lime import submodular_pick
import matplotlib.pyplot as plt

%matplotlib inline
```

In [2]:
```
df = pd.read_excel(r'./Data/default of credit card clients.xls', index_col = 0)
df_columns = df.iloc[0, :]
df = df[1:]
df.head()
```

Out[2]:

ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	…	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	default payment next month
1	20000	2	2	1	24	2	2	-1	-1	-2	…	0	0	0	0	689	0	0	0	0	1
2	120000	2	2	2	26	-1	2	0	0	0	…	3272	3455	3261	0	1000	1000	1000	0	2000	1
3	90000	2	2	2	34	0	0	0	0	0	…	14331	14948	15549	1518	1500	1000	1000	1000	5000	0
4	50000	2	2	1	37	0	0	0	0	0	…	28314	28959	29547	2000	2019	1200	1100	1069	1000	0
5	50000	1	2	1	57	-1	0	-1	0	0	…	20940	19146	19131	2000	36681	10000	9000	689	679	0

5 rows × 24 columns

Preprocess Data

In [3]:
```
df_processed = df.copy()
for column in df.columns:
    df_processed[column] = pd.to_numeric(df[column])

df_processed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 30000 entries, 1 to 30000
Data columns (total 24 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   LIMIT_BAL                   30000 non-null  int64
 1   SEX                         30000 non-null  int64
 2   EDUCATION                   30000 non-null  int64
 3   MARRIAGE                    30000 non-null  int64
 4   AGE                         30000 non-null  int64
 5   PAY_0                       30000 non-null  int64
 6   PAY_2                       30000 non-null  int64
 7   PAY_3                       30000 non-null  int64
 8   PAY_4                       30000 non-null  int64
 9   PAY_5                       30000 non-null  int64
 10  PAY_6                       30000 non-null  int64
 11  BILL_AMT1                   30000 non-null  int64
 12  BILL_AMT2                   30000 non-null  int64
 13  BILL_AMT3                   30000 non-null  int64
 14  BILL_AMT4                   30000 non-null  int64
 15  BILL_AMT5                   30000 non-null  int64
 16  BILL_AMT6                   30000 non-null  int64
 17  PAY_AMT1                    30000 non-null  int64
 18  PAY_AMT2                    30000 non-null  int64
 19  PAY_AMT3                    30000 non-null  int64
 20  PAY_AMT4                    30000 non-null  int64
 21  PAY_AMT5                    30000 non-null  int64
 22  PAY_AMT6                    30000 non-null  int64
 23  default payment next month  30000 non-null  int64
dtypes: int64(24)
memory usage: 5.7+ MB
```

In [4]:
```
X = df_processed.iloc[:, :-1]
y = df_processed.iloc[:, -1]
```

In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 123, stratify = y)
```

Decision Tree

In [6]:
```
clf = DecisionTreeClassifier(min_samples_split = 5, random_state = 123)
clf.fit(X_train.to_numpy(), y_train.to_numpy().astype('int'));
```

In [7]:
```
predictions = clf.predict(X_test.to_numpy())
```

In [8]:
```
acc = accuracy_score(y_test.astype('int'), predictions)
print(f'Accuracy of the classifier is {round(acc*100, 2)}%')
```

```
Accuracy of the classifier is 73.53%
```

In [9]:
```
feature_importance = pd.DataFrame()
feature_importance['variable'] = X_train.columns
feature_importance['importance'] = clf.feature_importances_
feature_importance.sort_values(by = 'importance', ascending = False)
```

Out[9]:

	variable	importance
5	PAY_0	0.170402
4	AGE	0.068602
19	PAY_AMT3	0.068038
11	BILL_AMT1	0.067394
0	LIMIT_BAL	0.053364
22	PAY_AMT6	0.050416
16	BILL_AMT6	0.048948
17	PAY_AMT1	0.047209
12	BILL_AMT2	0.047005
13	BILL_AMT3	0.045698
18	PAY_AMT2	0.042928
21	PAY_AMT5	0.042763
20	PAY_AMT4	0.042759
14	BILL_AMT4	0.042125
15	BILL_AMT5	0.040260
6	PAY_2	0.039836
2	EDUCATION	0.019410
3	MARRIAGE	0.014648
7	PAY_3	0.013586
1	SEX	0.009436
9	PAY_5	0.009280
10	PAY_6	0.008701
8	PAY_4	0.007192

In [10]:
```
i = np.random.randint(0, X_test.shape[0])
def lime_explainer(clf, X_train, X_test, i, mode = 'classification', class_names = ['No Default', 'Default'], num_features = 5):
    predict_fn = lambda x:clf.predict_proba(x)
    explainer = LimeTabularExplainer(X_train.values,
                                     mode = mode,
                                     feature_names = X_train.columns,
                                     class_names = class_names,
                                     discretize_continuous = True,
                                     verbose = True)
    explanation = explainer.explain_instance(X_test.to_numpy()[i+1],
                                             predict_fn,
                                             num_features = num_features)

    explanation.show_in_notebook()
```

In [11]:
```
lime_explainer(clf, X_train, X_test, i)
```

```
Intercept 0.41687851656740516
Prediction_local [0.30943478]
Right: 0.0
```

Random Forest

In [12]:
```
clf = RandomForestClassifier(n_estimators = 1000, n_jobs = -1, verbose = True)
clf.fit(X_train.to_numpy(), y_train.to_numpy().astype('int'))
```

```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:    0.1s
[Parallel(n_jobs=-1)]: Done 176 tasks      | elapsed:    1.0s
[Parallel(n_jobs=-1)]: Done 426 tasks      | elapsed:    2.5s
[Parallel(n_jobs=-1)]: Done 776 tasks      | elapsed:    4.5s
[Parallel(n_jobs=-1)]: Done 1000 out of 1000 | elapsed:    5.8s finished
RandomForestClassifier(n_estimators=1000, n_jobs=-1, verbose=True)
```

Out[12]:
```
RandomForestClassifier(n_estimators=1000, n_jobs=-1, verbose=True)
```

In [13]:
```
predictions = clf.predict(X_test.to_numpy())
```

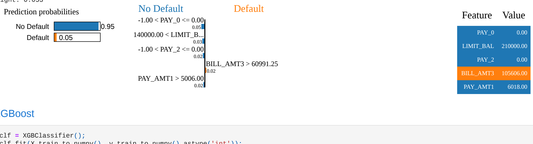
```
[Parallel(n_jobs=12)]: Using backend ThreadingBackend with 12 concurrent workers.
[Parallel(n_jobs=12)]: Done  26 tasks      | elapsed:    0.0s
[Parallel(n_jobs=12)]: Done 176 tasks      | elapsed:    0.0s
[Parallel(n_jobs=12)]: Done 426 tasks      | elapsed:    0.1s
[Parallel(n_jobs=12)]: Done 776 tasks      | elapsed:    0.2s
[Parallel(n_jobs=12)]: Done 1000 out of 1000 | elapsed:    0.3s finished
```

In [14]:
```
acc = accuracy_score(y_test.astype('int'), predictions)
print(f'Accuracy of the classifier is {round(acc*100, 2)}%')
```

```
Accuracy of the classifier is 81.69%
```

In [15]:
```
lime_explainer(clf, X_train, X_test, i)
```

```
[Parallel(n_jobs=12)]: Using backend ThreadingBackend with 12 concurrent workers.
[Parallel(n_jobs=12)]: Done  26 tasks      | elapsed:    0.0s
[Parallel(n_jobs=12)]: Done 176 tasks      | elapsed:    0.0s
[Parallel(n_jobs=12)]: Done 426 tasks      | elapsed:    0.1s
[Parallel(n_jobs=12)]: Done 776 tasks      | elapsed:    0.2s
[Parallel(n_jobs=12)]: Done 1000 out of 1000 | elapsed:    0.3s finished
Intercept 0.37366226474485567
Prediction_local [0.2738028]
Right: 0.053
```



XGBoost

In [16]:
```
clf = XGBClassifier();
clf.fit(X_train.to_numpy(), y_train.to_numpy().astype('int'));
```

```
c:\users\sangh\desktop\quantiphi\week 7\extra assignments\extra assignment 3\extra_assn_3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[11:04:20] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

In [17]:
```
predictions = clf.predict(X_test.to_numpy())
```

In [18]:
```
acc = accuracy_score(y_test.astype('int'), predictions)
print(f'Accuracy of the classifier is {round(acc*100, 2)}%')
```

```
Accuracy of the classifier is 81.75%
```

In [19]:
```
lime_explainer(clf, X_train, X_test, i)
```

```
Intercept 0.21357167472395927
Prediction_local [0.1461547]
Right: 0.041267477
```